A review on the detection of social circles in Facebook egonets

Louis de Vitry, Cynthia El-Hayek, Maximilien Barbe

***Abstract*— This paper reviews the use of community detection algorithms in social networks to automatically identify users social circles (ex. family, college friends) in social networks. With hierarchically nested and overlapping ground truth circles (hand labeled), we pose the problem as a multi-membership node clustering problem on a users ego-network, a network of connections between her friends. We first establish a model baseline for this problem, consisting of basic algorithms used in community detection: connected components, Girvan-Newman, greedy modularity optimization and k-clique. Then, we outline the current state of the art algorithm and implement it from scratch in Python (no publicly available code exists). Furthermore, as the evaluation of predicted communities is far from trivial for this problem, we give an exhaustive and comparative summary of methods used to this day. Finally, we evaluate and discuss the performance of all these algorithms on the standard dataset for this task, Facebook egonets. We show that the k-clique is the best algorithm among the baseline and that although theoritically powerful, the state of the art model is not robust to missing data. Finally, we suggest improvement directions for this task. Beside the review, our contribution to this problem is an open-source python package, containing the model baseline, the state of the art model, along with a complete evaluation module.**

I. Introduction

Detecting communities is a fundamental problem in the study of networks as strong community structure [1] is one of the main characteristics of a social network. In a purchasing behavior network, identifying customers with similar interest and characteristics can lead to better recommendation systems. In social networks, friend suggestion and consistent newsfeed is derived from community detection [2]. Also, with the huge volume of data and the recent scandals, privacy management has become a priority. Categorizing a user's list of connections into different circles gives better control over what to share with each circle. A good example is Instagram's new feature close friends.

Therefore motivated by the above reasons, we propose to work on automatically detecting social circles of Facebook users by implementing and evaluating basic algorithms used in community detection and the current state of the art algorithm.

II. Related work

The problem of detecting social circles in an ego network is certainly not a novel task and has been widely addressed in the literature, as several research and experiments have already been conducted. Wang et al. (2013) [3] proposed a link community algorithm that identify circles that are densely connected and whose members also share common properties. This approach overcomes some of the limitations of previous work such as failing to detect circles of small size [4], having a limit to the number of circles that can be detected [5] or requiring the number of circles to be predefined [6]. The authors first reconstruct ego networks to recover the missing relations and then give a new similarity definition of the edges and cluster edges by link community algorithm. Finally, they label circles to explain what features caused these circles emergence. In [2], the authors also took into consideration in their approach the similarity between users that belong to the same community. The authors proposed a joint probabilistic model for combining link and node features for community detection. They first build a model with the network links only. Then, they created a new feature-based weighted network whose edge weight is the node feature similarity between two nodes. Then, they merged the original network and the created network. If two nodes have a strong similarity, the original link between the two nodes will be strengthened, otherwise it will be weakened. An expectation-maximization algorithm (EM) was employed for the optimization.

Many of the resulting algorithms can handle a broad range of problems, which makes their study even more crucial. In the table below, we list major techniques used to tackle this problem.

Models	Type
K-means clustering	ML
Multi-assignment clustering	ML
Link clustering	Network
Clique Precolation	Network
Mixed membership stochastic block models	Network + ML
Block-LDA	Network + ML
Low-rank embedding	Network + ML
State-of-the-art	Network + ML

Fig. 1. Overview of the algorithms used in the litterature

where Type refers to the underlying theoretical foundation of the algorithm, with ML denoting the use of Machine Learning.

III. Data set and task descriptions

In this work, we used only Facebook anonimized data. The dataset was provided by a Kaggle competition [7] and the SNAP repository [8]. It consists of 111 ego networks each ego network provided in a separate file and representing a user and his alters (friends). In each file, one line refers to

one alter id and a list of all the friends id that he has in common with the ego in question.

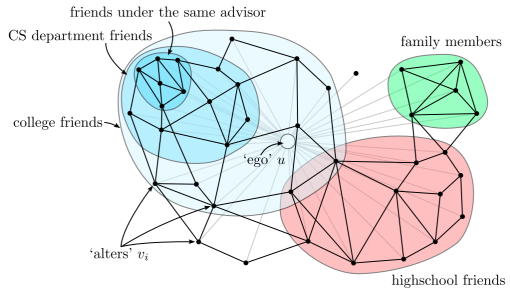

Fig. 2. Egonet with ground-truth circles

We furthermore had access to multiple files containing users features (mainly consisting of personal information such as first name, employer, hometown, languages, birthday, etc.) and to hand-labeled ground-truth communities for 61 ego networks (i.e. circles each consisting of a subset of friends of the ago network sharing common attributes like university, sports team or else). These ground truth information were manually provided by users who were asked to do so. For matters of confidentiality, the values of this features were replaced by numbers.

A. Exploratory Data Analysis

As we can see from the distributions below, we observe that the underlying topology of the community structure is complex and heterogeneous. It therefore legitimates the need to design flexible and robust algorithms to deal with such structures.

Similarly to the Exploratory Data Analysis phase in Machine Learning applications, we also need a more thorough analysis of our data (see future work).

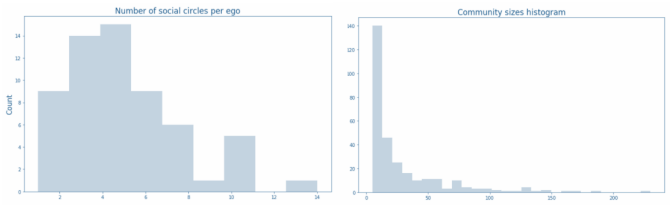

Fig. 3. Circles and community tructures

B. Feature engineering

In our dataset, a lot of users personal information and which might constitute potential features (i.e. university, location, religion, political interests, etc.) presented a high number of missing values (more than 50%) making them non-useful. Since methods of filling missing values were not consistent except for birthdays, it has been decided to get rid of all the variables presenting more than 30% of missing values which reduces our number of features from 56 to 2 features: gender and last time. Pairwise node similarity features were generated based on the gender and last name users information. The main motivation behind using the last name is to know if two user are of the same family.

IV. Algorithms

In this section, we will present an overview of the algorithms landscape. First, we implement a baseline consisting of two foundationals techniques of graph clustering: link clustering and clique percolation, and confront them with the ground truths circles. These algorithms work only with the network structure. We therefore implement the current state of the art for this task, which is a generative model for friendship based on node features.

A. Link clustering

Girvan Newman Community Detection

The Girvan-Newman algorithm follows a divisive method and looks for the edges in the network that are responsible for connecting many pairs of other nodes. It progressively removes edges from the graph (the ones with the highest betweenness centrality) at each step which exposes the result as a dendrogram. At each step of the algorithm, betweenness scores are re-calculated [9][10]. The method that we applied in this projects uses shortest path as a betweenness centrality measure.

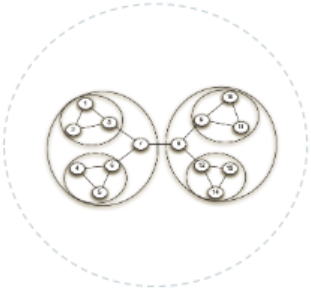

Fig. 4. Girvan Newman scheme

Although this method works really well because of the recalculation of the betweenness centrality at each iteration, it fails to detect overlapping communities.

Greedy Modularity Optimization

This algorithm is based on the idea of modularity introduced by Girvan Newman community structure to test whether a particular division is meaningful and it is defined as:

$$Q = \sum_i e_{ii} - a_i^2$$

with e being a symmetric matrix whose element e_{ij} is the fraction of all edges in the network that link vertices in community i to vertices in community j and a_i the fraction of all ends of edges that are attached to vertices in group i.

If a particular division gives no more within-community edges than would be expected by random chance Q will be 0 and values other than 0 indicate deviations from randomness. Significant community structure appears when Q is greater than 0.

Greedy modularity optimization begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists [11].

The method we applied in this project uses a more optimized version of the greedy modularity optimization suggested in [12] and that is more convenient for extremely large networks. We implemented it with Networkx.

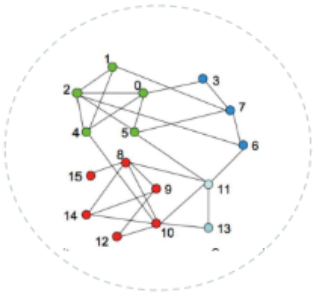

Fig. 5. Greedy modularization scheme

Although this method is very efficient for extremely large networks, ans similarly to Girvan-Newman method, it remains unable to detect overlapping communities.

B. Clique percolation

A k-clique-community is defined as a union of all complete subgraphs of size k (k-cliques) that can be reached from each other through a series of adjacent k-cliques, adjacency meaning sharing k-1 nodes. This definition is based on the idea that the members of a community can be reached through well connected subsets of nodes [13]. In this project we tested different values of k and obtained optimal results for k equal to 7.

Fig. 6. k-clique scheme

The main advantage of this method is that it can detect overlapping communities as opposed to Girvan-Newman and Greedy modularity optimization that can only detect disjoint communities.

Connected Components Community Detection
This last method that we used is based on the simple idea of detecting connected components in a graph. Each connected component subgraph in which each 2 vertices are connected by a path and which is connected to no additional vertices in the graph - is considered as a community. Of course and by definition, the detected communities do not overlap. This method is expected to have a lower performance compared to the other baselines given that it does not really reflect the definition of communities in a real life setting where communities have members that are connected to each other.

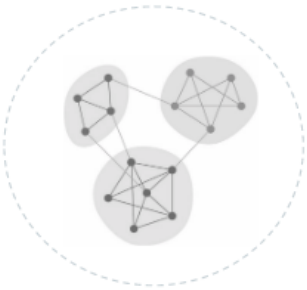

Fig. 7. Connected components scheme

All the methods that we described above and used as baselines in this project are limited in a sense that they cant incorporate non graph-based features such as personal information available for each user (node).

C. State of the art

A generative model for friendship in social circles
Here we explain in details the state of the art model [5] used for community detection in social networks. It is an unsupervised model whose core is a generative model for friendship in social circles, that defines the probability that two nodes forming an edge as

$$p((x,y) \in E) \propto \exp\Bigg(\sum_{C_k \supseteq \{x,y\}} \langle \phi(x,y), \theta_k \rangle - \sum_{C_k \not\supseteq \{x,y\}} \alpha_k \langle \phi(x,y), \theta_k \rangle \Bigg) \quad (1)$$

with:

- θ_k: vector describing community C_k
- α_k: regularization term that penalizes for larger k
- $\phi(x,y)$: similarity vector between two nodes x, y

Similarity vector
The quantity $\phi(x,y)$ is a similarity vector between nodes x and y, that integrates the nodes features. The construction of such features is explained in data set and task section.

Log likelihood
Given that the edges between nodes, each edge represented

as $e = (x, y)$, are generated independently, the probability of an ego-network, G, can be written down as the product of positive evidence for all circles and negative evidence against all circles. For given parameters $(\theta_k)_{k=1,\dots,K}$ and $(\alpha_k)_{k=1,\dots,K}$, we formulate the log-likelihood as

$$l_\Theta(G, \mathcal{C}) = \sum_{e \in E} \Phi(e) - \sum_{e \in E} \log(1 + \exp \Phi(e))$$

with

$$\Phi(e) = \sum_{C_k \in \mathcal{C}} \big(\delta(e \in C_k) - \alpha_k \delta(e \notin C_k)\big) \langle \phi(e), \theta_k \rangle$$

Coordinate ascent
By considering the circles $\mathcal{C}$ as latent variables, we can find $\Theta = (\{\theta_k\}_{k=1}^K, \{\alpha_k\}_{k=1}^K)$ using coordinate ascent on Θ and $\mathcal{C}$

$$\begin{aligned} \mathcal{C}^t &= \text{argmax}_{\mathcal{C}} l_{\Theta^t}(G; \mathcal{C}) \\ \Theta^{t+1} &= \text{argmax}_{\Theta} l_\Theta(G, C^t) - \lambda \Omega(\theta) \end{aligned} \tag{2}$$

where $\Omega(\theta)$ is a regularizaton parameter (see L-BFGS solver).

The former equation can be solved using quadratic pseudo-boolean optimization (QPBO) and the latter with L-BFGS.

The original and full algorithm of the state is given below:

ALGORITHM 1: Predict complete circles with hyperparameters λ, K.

Data: ego-network $G = (V, E)$, edge features $\phi(e) : E \to \mathbb{R}^F$, hyperparameters λ, K
Result: parameters $\hat{\Theta} := \{(\hat{\theta}_k, \hat{\alpha}_k)\}^{k=1\dots K}$, communities $\hat{\mathcal{C}}$
initialize $\theta_k^0 \in \{0, 1\}^F$, $\alpha_k^0 := 1$, $C_k := \varnothing$, $t := 0$;
repeat
  for $k \in \{1 \dots K\}$ **do**
    $\mathcal{C}_k^t := \text{argmax}_C \sum_{(x,y) \in V \times V} E^k_{(x,y)}(\delta(x \in C), \delta(y \in C))$;
    // using QPBO
  end
  $\Theta^{t+1} := \text{argmax}_\Theta \, l_\Theta(G; \mathcal{C}^t) - \lambda \Omega(\theta)$;
  // using L-BFGS
  $t := t + 1$;
until $C^{t+1} = \mathcal{C}^t$;

Fig. 8. State of the art algorithm

Quadratic pseudo-boolean optimization
For a given set of parameters $\theta \in \Theta$, the problem of maximizing the log-likelihood with respect to the communities can be arguably solved by a quadratic pseudo-boolean optimization (QPBO) on a Markov Random Field, according to Boros et Hammer (2002) [14].

In a nutshell, the QPBO algorithm finds the binary variables (whether or not a node is assigned to a particular community C_k) that maximize the sum of continuous-valued energy functions over pairwise terms (the energies of each possible edge) based on roof duality.

McAuley et Lescovek (2013) [5] provide the retranscription of the problem in the general form of pairwise QPBO (see implementation notes)

$$C_k = \text{argmax}_{\mathcal{C}} \sum_{(x,y) \in V \times V} E^k_{(x,y)}\big(\delta(x \in C), \delta(y \in C)\big)$$

where $E^k_{(x,y)} : \{0, 1\}^2 \mapsto R$ are the energy functions corresponding to the pairwise Markov Random Fields. In our setting, binary variables encode the memberships of nodes to a particular circle C_k, yielding four distinct cases, to which we attribute a certain energy level, depending on desirability

1) Low energy: an edge appears outside the circle
2) High energy: a non-edge appears outside the circle
3) Low energy: a non-edge appears inside the circle
4) High energy: an edge appears inside the circle

The exact derivation of $E^k_{(x,y)}$ (refer to the original publication [5]) furthermore considers the feature similarity $\phi(e)$ and the community parameter θ_k.

Ultimately, the QPBO algorithm [15] works by minimizing possibly non-submodular dual with graph cuts. Afterwards, boolean labels for each node are recovered from their assignments to "source" and "sink" sets.

We used a Python package thinqpbo [16] that provides an API of the C++ package developed by Vladimir Kolmogorov [17]. However, the documentation of these packages being rather limited, our use of the API remains rudimentary (see Future work).

Implementation notes:

- "Outside the circle" also includes the case where one of the nodes appears inside the circle but the other does not, that is where the pair of nodes crosses the circle.
- We negated the energies because QPBO solve minimization problems.
- The QPBO was applied in a random order.

L-BFGS solver for log-likelihood
After the newly found communities $\mathcal{C}^t$ were computed, we used scipy L-BFGS solver [18] to minimize the negative regularized log-likelihood

$$\Theta^{t+1} = \text{argmax}_\Theta l_\Theta(G, C^t) - \lambda \Omega(\theta)$$

where

$$\Omega(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{|\theta_k|} | \, \theta_{ki} \, |$$

acts as an l_1 regularization, therefore producing sparse and readily interpretable parameters.

Additionally, we provided the L-BFGS the partial derivatives, given by

$$\frac{\partial l}{\partial \theta_k} = \sum_{e \in V \times V} -d_e(k)\phi(e)_k \frac{e^{\Phi(e)}}{1+e^{\Phi(e)}} + \sum_{e \in E} d_k(e)\phi(e)_k - \frac{\partial \Omega}{\partial \theta_k}$$

$$\frac{\partial l}{\partial \alpha_k} = \sum_{e \in V \times V} \delta(e \notin C_k)\,\langle \phi(e), \theta_k \rangle \frac{e^{\Phi(e)}}{1+e^{\Phi(e)}} - \sum_{e \in E} \delta(e \notin C_k)\,\langle \phi(e), \theta_k \rangle$$

The initial solution of the solver $x_0 = ((\alpha_k)_{k=1}^K, (\theta_k)_{k=1}^K)$ is defined as follows

- $(\theta_k)_{k=1}^K \in \{0,1\}^F$ initialized randomly
- $\alpha_k = 1,\ \forall k \in \{0, ..., K\}$

Implementation notes:

- The scipy solver minimizes the function so we negated the log-likelihood.
- Empirically, a maximum number of iterations set at 10 is enough to obtain a consequent increase of log-likelihood.

Computing hyperparameters λ and K
In the original publication [5] the authors mention they used the Bayesian Information Criterion to select the number of communities. Being fundamentally limited on the number of communities (see Results), we did not optimized K. Furthermore, they note that the regularization parameter λ has little to no effect when changed. (see future work).

V. Evaluation

A. Metrics

Social circles detection needs an evaluation metric to assess the performance of an algorithm given ground-truth circles. In this context, the evaluation consists of two steps

1) Defining a similarity score $s(C, C^*)$ between a predicted circle C and a ground-truth circle C^*.
2) Aligning the set of predicted circles to the set of ground-truth circles so as to maximize the total sum of similarities.

Similarity metrics
We list in the table below the most common similarity measures used in the literature.
We then compute the scores between each pairs (C, C^*) and store them in a matrix. The resulting score matrix is non-necessarily square as the set of predicted and true circles can differ in size.

It should be noted that both the F1 score and BER do not induce a penalty for large circles (as they are kind of normalized metric). This explains why most recently, the edit distance has gained in popularity.

Similarity measure	Formula
Precision	$\frac{\lvert C \cap \hat{C} \rvert}{\lvert C \rvert}$
Recall	$\frac{\lvert \hat{C} \cap C \rvert}{\lvert \hat{C} \rvert}$
F-measure	$2\frac{\text{precision}(C,\hat{C}) \cdot \text{recall}(C,\hat{C})}{\text{precision}(C,\hat{C}) + \text{recall}(C,\hat{C})}$
Balanced Error Rate	$\frac{1}{2}(\frac{\lvert C \setminus \hat{C} \rvert}{\lvert C \rvert} + \frac{\lvert \hat{C} \setminus C \rvert}{\lvert \hat{C} \rvert})$
Edit distance	$\lvert C \cup \hat{C} \rvert - \lvert C \cap \hat{C} \rvert$

Fig. 9. Similarity metrics

Edit distance: The edit distance is a custom loss coming from the original Kaggle competition Learning Social Circle, from which we took our datasets. It has four basic edit operations, all of them at cost 1

- Adding a user to an existing circle
- Creating a circle with one user
- Removing a user from a circle
- Deleting a circle with one user

This metric is mainly useful because it actually reflects how far we are from the ideal solution (ground truth circles).

Ground truth labels provided by facebook data were incomplete meaning that the true communities dont partition the whole egonet (i.e. there are nodes belonging to no communities). As this is troublesome for some metrics, we created a community (circle) with all singled out nodes.

Aligning predicted communities to ground-truth circles
Once those features are computed and stored in a cost matrix, we proceed with aligning the communities the communities C to their ground-truth counterpart. Mathematically, there are two methods:

1) Best Matching evaluation measure: For every detected circle, we find its best ground-truth match and compute the performance (we do this reciprocally as well). The final evaluation function is the average of the two performance measures:

$$E_b(\mathcal{C}, \bar{\mathcal{C}}) = \frac{1}{2|\bar{\mathcal{C}}|} \sum_{\bar{C}_i \in \bar{\mathcal{C}}} \max_{C_j \in \mathcal{C}} s(\bar{C}_i, C_j) + \frac{1}{2|\mathcal{C}|} \sum_{C_j \in \mathcal{C}} \max_{\bar{C}_i \in \bar{\mathcal{C}}} s(\bar{C}_i, C_j)$$

   The main drawback of this metric is that frequently, several ground-truth circles are aligned to just one predicted circle or vice versa. Furthermore, there is no penalization of non-aligned predicted community or ground truth.
2) Hungarian Matching evaluation measure: The alignment is defined as an optimal correspondence via linear assignment, found by means of the Hungarian algorithm:

$$E_h(\mathcal{C}, \bar{\mathcal{C}}) = \max_{f:\mathcal{C}\to\bar{\mathcal{C}}} \frac{1}{|f|} \sum_{C\in \mathrm{dom}(f)} (1 - s(C, f(C))$$

There are no cases of single-to-multiple circles alignment. However, the Hungarian algorithm can suffer from degenerate optimal performance; when it match small predicted circles with low cost, thus leaving possibly large unaligned circles.

In each case, the final evaluation measure is the average over the egonets in the dataset.

VI. Results

We believe that the edit distance is the most complete and accurate evaluation measure of the ones described above, as it is a global measure between sets of circles, it does not consider single-to-multiple circles alignments and it does not lead to degenerate optimal performance.

Baseline results

All the baseline models but the k-clique algorithm depend on hyperparameters, we therefore implement as is. As for the k-clique, we find the optimal hyperparameter k by running the algorithms for various of k. We report the results in the boxplot below:

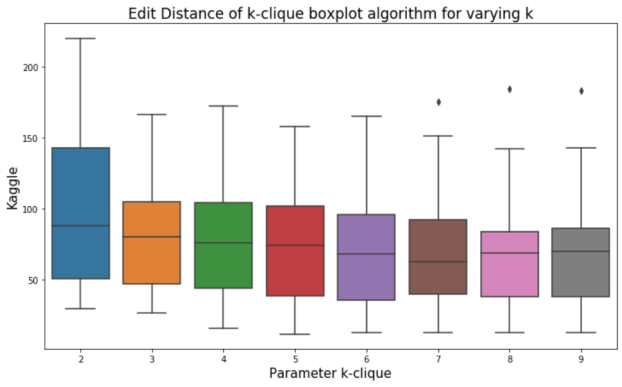

Fig. 10. Boxplot of the edit distances of k-clique algorithm for various k (taken over all egonets)

We choose $k = 7$ as it minimizes the edit distance over all egonets. Then we run all the models of our baseline and display the results below in terms of edit distance as previously explained.

As we can see, Girvan-Newman and Greedy Modularity Optimization have similar performances having almost the same spread and the median (90) of the edit distance distribution. The 7-clique method performs better than the aforementioned methods having a lower median (75) and a smaller spread of the edit distance distribution. However, the connected components method has a significantly larger distribution spread than the other methods but a similar median to Girvan-Newman and Greedy Modularity

State of the art

As expected, the algorithms most often than not produce

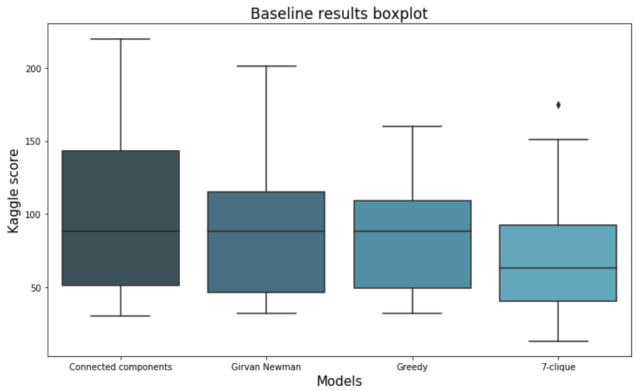

Fig. 11. Boxplot of the edit distances of our model baseline (taken over all egonets)

degenerate results (i.e. empty communities and communities consisting of the whole egonet mostly). The main reasons for this failure lie in the dimensionality F of the features $\phi(x, y)$. More precisely, the limits of this approach on our dataset are characterized by:

- Robusteness: The iterative algorithm is heavily influenced by the initialization of the parameters Θ_0. Two differents initialization will produce utterly different results.
- Dot product For a community C_k and nodes x and y, $\langle \phi(x, y), \theta_k \rangle$ has far less discriminative power than it would if F was higher.
- Number of possible communities K is upper-bounded by the number of possible arrangements of Θ_0 (i.e. $2^F = 4$ in our case).

Despite the aforementioned limits, it should be noted that the algorithm almost always yield strong log-likelihood increase.

In the original publication[5], the authors point out that they manually constructed facebook egonets with the full features, but they did not release it publicly. It explains why they got so good results with facebook features.

VII. Future work

Exploratory Data Analysis

A more careful and attentive analysis of the social graph (typically mean degree, connectedness, degree distribution...) would be helpful to further understand the egonet topologies.

Better tests

As we see from our results, it performed poorly with missing data. To analyze how robust is this model, we could vary the percentage of missing values (roughly correlates to the dimension of our feature similarity vector $\phi(x, y)$) and see the effect on the score.

Optimize code

Though we performed decent module testing of our code,

time limits prohibited us to run this package on other datasets. For robustness purposes, the code would benefit to be tested against google+ dataset. Beside data related issues, several speed optimizations can be added to the code:

1) Use symmetric property of the energy matrices
2) Handle the feature matrix properly
3) Clever vectorization

QPBO

Implementing a well-documented python version of the QPBO algorithm from scratch (as described in [15]) would be very useful. This could give a much bigger control over the node-communities assignments (including possible penalty for semi-supervised learning) as well as statistics and diagnosis tools of the resolution. We nonetheless provide the normalization part (python).

Furthermore, the QPBO seems to be a natural framework for clustering. This framework has possibly broader applications outside of this specific generative model optimization.

Hyperparameters selection

Once we test our algorithm against rich datasets, a module should take care of the hyperparameter optimization. Moreover, the use of a Dirichlet Process to discover the number of clusters should be envisioned, as the BIC can be prone to overfitting problems.

Other techniques

Several other techniques which use node features reached good results and should be further explored as well. They are Block-Latent Dirichlet Allocation and low-rank embedding.

Python implementation of those models[19] (there exists a python implementation of block-LDA but as the author points out, it is a little rough on the edges and we did not find an implementation of the low-rank embedding).

REFERENCES

[1] Girvan, Michelle, and Mark EJ Newman. "Community structure in social and biological networks." Proceedings of the National Academy of Sciences99.12 (2002): 7821-7826

[2] Zhang, F., Li, J., Li, F., Xu, M., Xu, R., He, X. (2015, January). Community detection based on links and node features in social networks. In International Conference on Multimedia Modeling (pp. 418-429). Springer, Cham.

[3] Wang, Y., Gao, L. (2013). An Edge-based Clustering Algorithm to Detect Social Circles in Ego Networks. JCP, 8(10), 2575-2582.

[4] Stephan Gnnemann, Brigitte Boden , Thomas Seidl, Finding density-based subspace clusters in graphs with feature vectors, Data Mining and Knowledge Discovery, 2012, vol.25, pp. 243-269

[5] Julian McAuley and Jure Leskovec, Learning to discover social circles in ego networks, Neural Information Processing Systems, 2012, pp. 548-556

[6] T. Yoshida, Toward finding hidden communities based on user profiles, In Proceeding of the IEEE International Conference on Data Mining Workshops, 2010. pp. 380-387

[7] "learning social circles in networks", Kaggle, 24 Jan 2019, https://www.kaggle.com/c/learning-social-circles/data

[8] add link here

[9] NetworkX Developers, "networkx.algorithms.community.centrality.girvan_newman" NetworkX, 24 Jan 2019, https://networkx.github.io/documentation/latest/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html

[10] Newman, Mark EJ, and Michelle Girvan. "Finding and evaluating community structure in networks." Physical review E 69.2 (2004): 026113.

[11] NetworkX Developers, "networkx.algorithms.community.modularity_max.greedy_modularity_communities" NetworkX, 24 Jan 2019, https://networkx.github.io/documentation/latest/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html

[12] Clauset, Aaron, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks." Physical review E 70.6 (2004): 066111.

[13] uncovering the overlapping community structure of complex networks in nature and society.

[14] Boros, Endre L. Hammer, Peter. (2002). Pseudo-Boolean Optimization. Discrete Applied Mathematics. 123. 155-225. 10.1016/S0166-218X(01)00341-9.

[15] http://wwwpub.zih.tu-dresden.de/ cvweb/publications/papers/2007/PAMI07-QPBO.pdf

[16] https://pypi.org/project/thinqpbo/

[17] http://pub.ist.ac.at/ vnk/software.html

[18] "optimization and root finding", Scipy, 25 Jan 2019, https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_l_bfgs_b.html

[19] https://github.com/rbalasub/jigsaw/blob/master/README